

Chapter 4

เปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ENGCE117 Computer Programming for Computer Engineer
Kittinan Noimanee · Computer Engineering · RMUTL

เนื้อหา

1. การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์
2. Simple Linked List

การเปลี่ยนตำแหน่งการชี้ ของพอยน์เตอร์

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

- เพื่อให้เข้าใจ Function ที่สามารถเปลี่ยน Pointer ให้ไปชี้ที่อื่น ๆ ได้
- เพื่อให้เข้าใจการนำข้อมูลที่ Pointer ชี้อยู่ภายนอก มาใช้งาน ภายใน Function ได้
- เข้าใจการใช้ Pointer ชี้อ้างอิง และ Pointer ที่ชี้ Pointer ได้

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

```
3 void go( int **p, int *z ){
4     *p = z;
5     printf("%d %p %p\n", **p, *p, p);
6 }
7
8 int main(){
9     int *a, b = 10, c = 20;
10    go(&a, &b);
11    printf("%d %p %p\n", *a, a, &a);
12    printf("-----\n");
13    go(&a, &c);
14    printf("%d %p %p\n", *a, a, &a);
15
16    return 0;
17 }
```

```
10 00000000000022FD74 00000000000022FD78
10 00000000000022FD74 00000000000022FD78
-----
20 00000000000022FD70 00000000000022FD78
20 00000000000022FD70 00000000000022FD78
```

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

- Pointer 1 ระดับถูกประกาศใน main
- แต่ Function ย่อยสามารถเปลี่ยนที่ชี้ของ Pointer ให้ไปชี้ยังที่อื่นได้ด้วย
- ในการนำมาใช้งาน หลังจากที่ Function ย่อยเปลี่ยนที่ชี้ของ Pointer เสร็จแล้ว ใน main ก็สามารถเรียกใช้ Pointer นั้นๆ ได้เลย (เพราะเปลี่ยนที่ชี้แล้ว)

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

```
3 void go( int **p, int *z ){
4     *p = z;
5     printf("%d %p %p\n", **p, *p, p);
6 }
7
8 int main(){
9     int *a, b = 10, c = 20;
10    go(&a, &b);
11    printf("%d %p %p\n", *a, a, &a);
12    printf("-----\n");
13    go(&a, &c);
14    printf("%d %p %p\n", *a, a, &a);
15
16    return 0;
17 }
```

ตอนแรกยังไม่ได้ชี้อะไร

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

```
3 void go( int **p, int *z ){
4     *p = z;
5     printf("%d %p %p\n", **p, *p, p);
6 }
7
8 int main(){
9     int *a, b = 10, c = 20;
10    go(&a, &b);
11    printf("%d %p %p\n", *a, a, &a);
12    printf("-----\n");
13    go(&a, &c);
14    printf("%d %p %p\n", *a, a, &a);
15
16    return 0;
17 }
```

ส่ง Pointer ไปหลายๆ เพื่อให้
ไปกำหนดตำแหน่งการชี้ที่
Function แทน

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

```
3 void go( int **p, int *z ){
4     *p = z;
5     printf("%d %p %p\n", **p, *p, p);
6 }
7
8 int main(){
9     int *a, b = 10, c = 20;
10    go(&a, &b);
11    printf("%d %p %p\n", *a, a, &a);
12    printf("-----\n");
13    go(&a, &c);
14    printf("%d %p %p\n", *a, a, &a);
15
16    return 0;
17 }
```

กำหนดจุดชี้ให้กับ *a เป็น
ที่เรียบร้อย โดยให้ชี้ไปยัง
ค่าของ b ดังนั้น a จึงเก็บ
Address ของ b เอาไว้

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 1: Pointer 1 ระดับ

```
3 void go( int **p, int *z ){
4     *p = z;
5     printf("%d %p %p\n", **p, *p, p);
6 }
7
8 int main(){
9     int *a, b = 10, c = 20;
10    go(&a, &b);
11    printf("%d %p %p\n", *a, a, &a);
12    printf("-----\n");
13    go(&a, &c);
14    printf("%d %p %p\n", *a, a, &a);
15
16    return 0;
17 }
```

เมื่อออกจาก Function ก็
พบว่าค่าของ *a ได้ถูก
กำหนดจุดชี้เป็นที่
เรียบร้อยแล้ว

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 2: Pointer 2 ระดับ

```
3 void go( int ***p, int **z ){
4     *p = z;
5     printf("%d %p %p %p\n", ***p, **p, *p, p);
6 }
7
8 int main(){
9     int *b = new int; *b = 10;
10    int *c = new int; *c = 20;
11    int **a;
12    go(&a, &b);
13    printf("%d %p %p %p\n", **a, *a, a, &a);
14    printf("-----\n");
15    go(&a, &c);
16    printf("%d %p %p %p\n", **a, *a, a, &a);
17    return 0;
18 }
```

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 2: Pointer 2 ระดับ

```
10 000000000004E7590 0000000000022FD78 0000000000022FD68
10 000000000004E7590 0000000000022FD78 0000000000022FD68
-----
20 000000000004E75B0 0000000000022FD70 0000000000022FD68
20 000000000004E75B0 0000000000022FD70 0000000000022FD68
```

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 2: Pointer 2 ระดับ

- Pointer 2 ระดับถูกประกาศใน main
- แต่ Function ย่อยสามารถเปลี่ยนที่ชี้ของ Pointer ให้ไปชี้ยังที่อื่นได้ด้วย
- ในการนำมาใช้งาน หลักจากที่ Function ย่อยเปลี่ยนที่ชี้ของ Pointer เสร็จแล้ว ใน main ก็สามารถเรียกใช้ Pointer นั้นๆ ได้เลย (เพราะเปลี่ยนที่ชี้แล้ว)

การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 2: Pointer 2 ระดับ

- ข้อสังเกตคือ การชี้ไปยัง Pointer อีกตัวหนึ่ง ตัวที่จะชี้จะต้องมีคีย์หรือระดับของ Pointer มากกว่าตัวชี้อยู่ 1 ตัวเสมอ

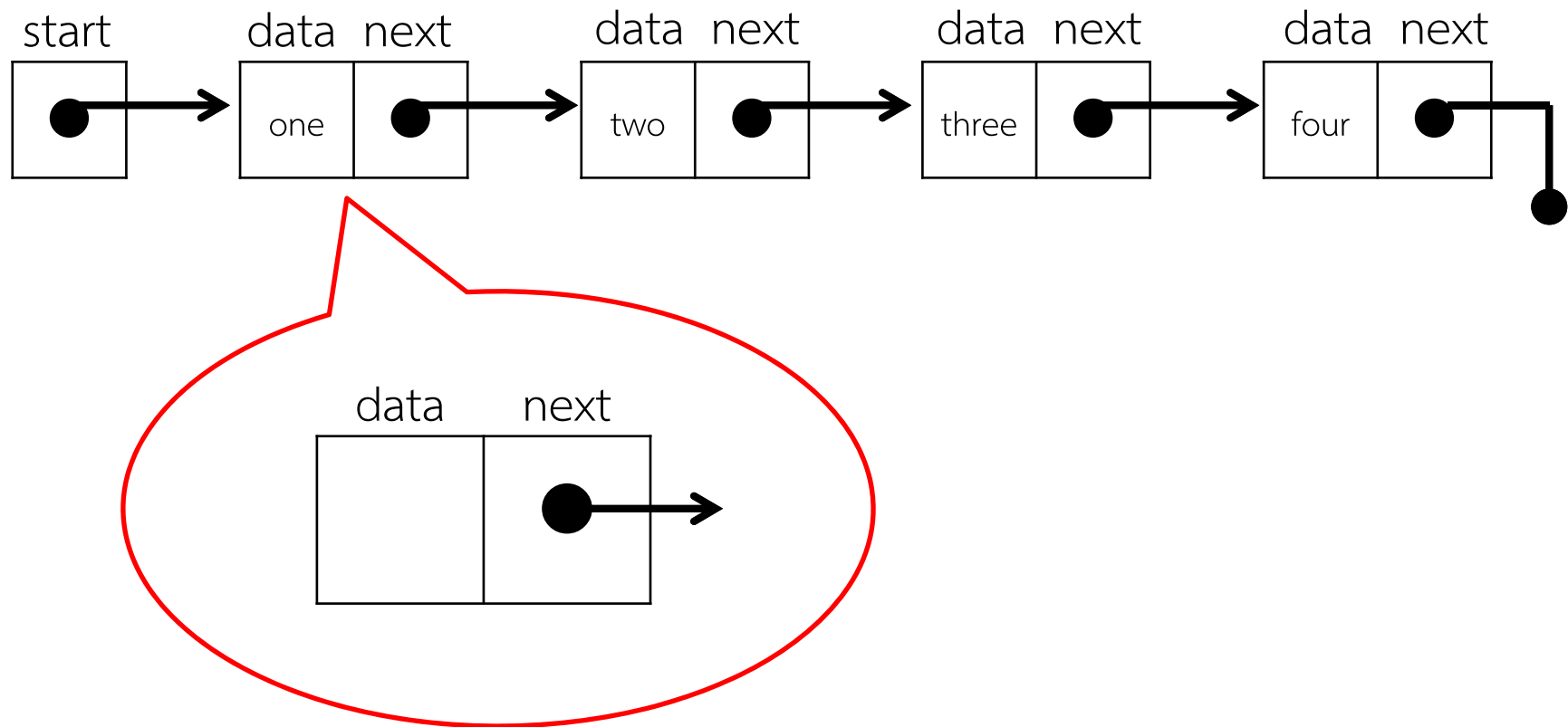
การเปลี่ยนตำแหน่งการชี้ของพอยน์เตอร์

ตัวอย่างที่ 2: Pointer 2 ระดับ

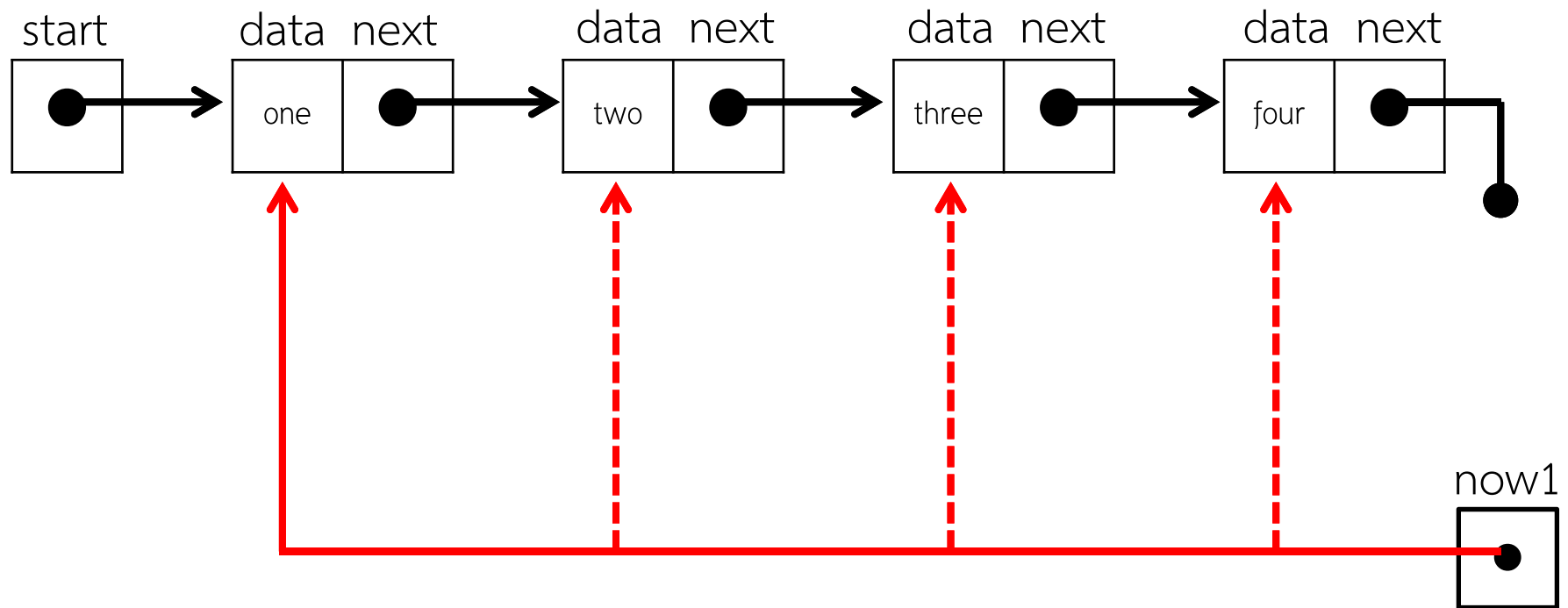
```
3 void go( int ***p, int **z ){
4     *p = z;
5     printf("%d %p %p %p\n", ***p, **p, *p, p);
6 }
7
8 int main(){
9     int *b = new int; *b = 10;
10    int *c = new int; *c = 20;
11    int **a;
12    go(&a, &b);
13    printf("%d %p %p %p\n", **a, *a, a, &a);
14    printf("-----\n");
15    go(&a, &c);
16    printf("%d %p %p %p\n", **a, *a, a, &a);
17    return 0;
18 }
```

Simple Linked List

Simple Linked List

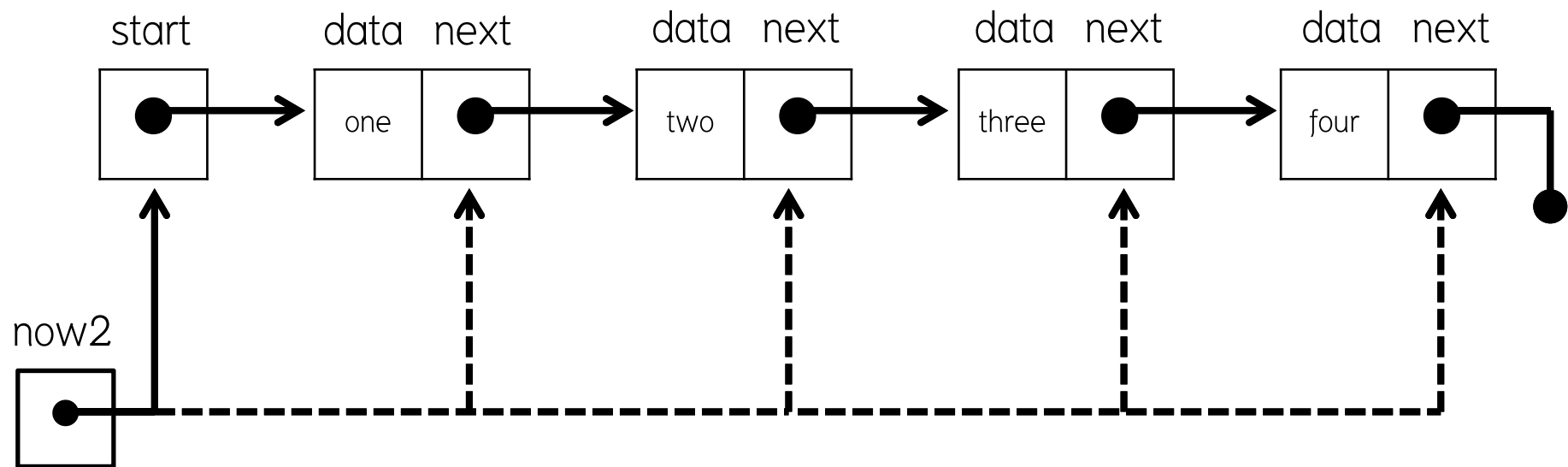


Simple Linked List



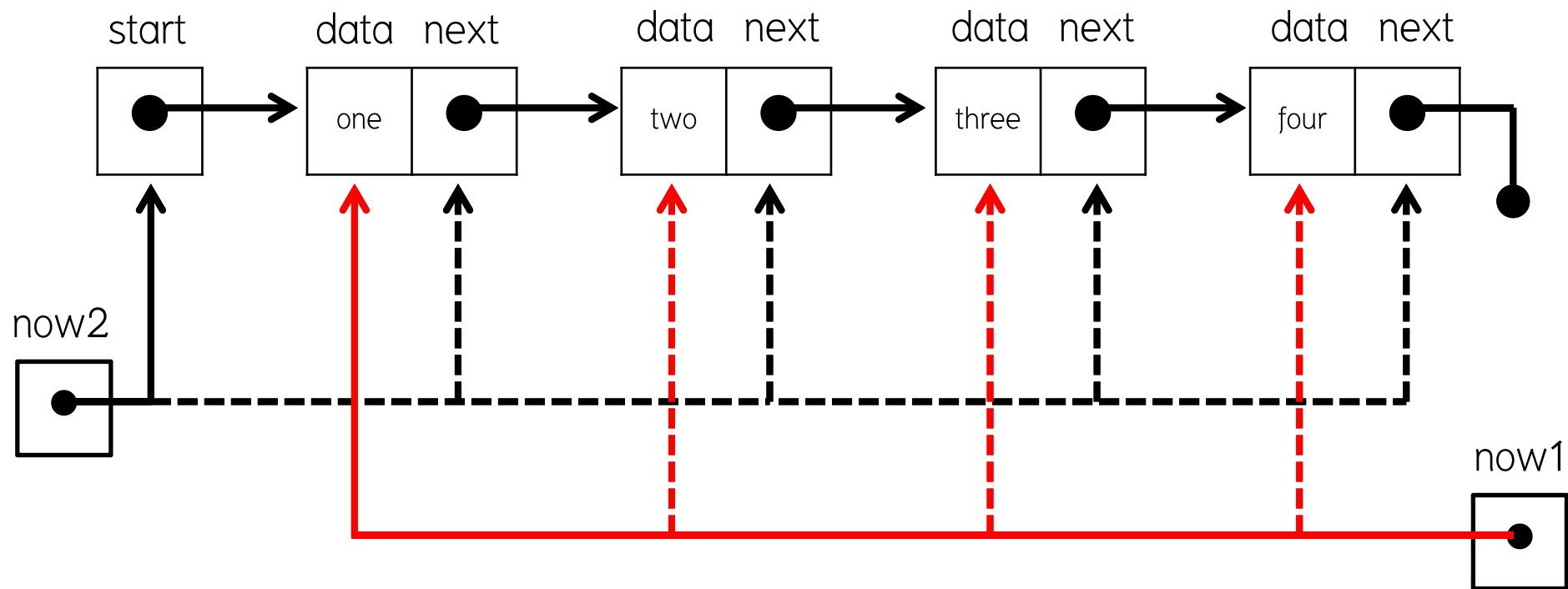
หมายเหตุ: now1 ใช้ชี้ Data

Simple Linked List



หมายเหตุ: now2 ใช้ชี้ตัวถัดไป (Next)

Simple Linked List



Simple Linked List

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct studentNode{
5      char name[20];
6      int age;
7      char sex;
8      float gpa;
9      struct studentNode *next;
10 };

```

Simple Linked List

```
12 void SaveNode(struct studentNode *child, char n[], int a, char s, float g);  
13
```

Simple Linked List

```
32 void SaveNode(struct studentNode *child, char n[], int a, char s, float g){  
33     strcpy(child->name, n);  
34     child->age = a;  
35     child->sex = s;  
36     child->gpa = g;  
37 }
```

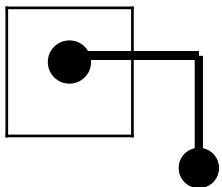
Simple Linked List

```
15 □ int main(){
16     struct studentNode *start, *now1, **now2;
17     → start = new struct studentNode;
18     → SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     → SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     → SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     → SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```


Simple Linked List

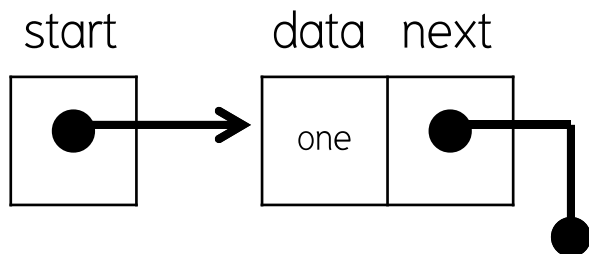
```
15 int main(){
16     struct studentNode *start, *now1, **now2;
17     start = new struct studentNode;
18     SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```

start



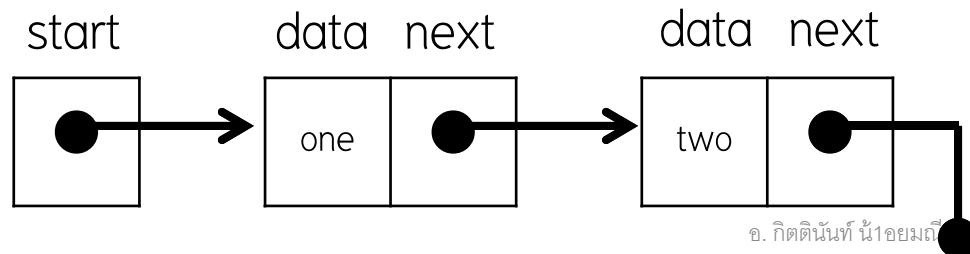
Simple Linked List

```
15 □ int main(){
16     struct studentNode *start, *now1, **now2;
17     start = new struct studentNode;
18     → SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```



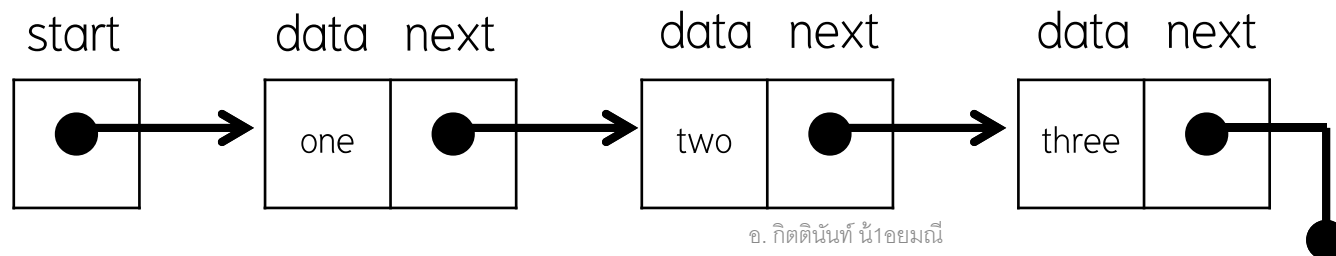
Simple Linked List

```
15 int main(){
16     struct studentNode *start, *now1, **now2;
17     start = new struct studentNode;
18     SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```



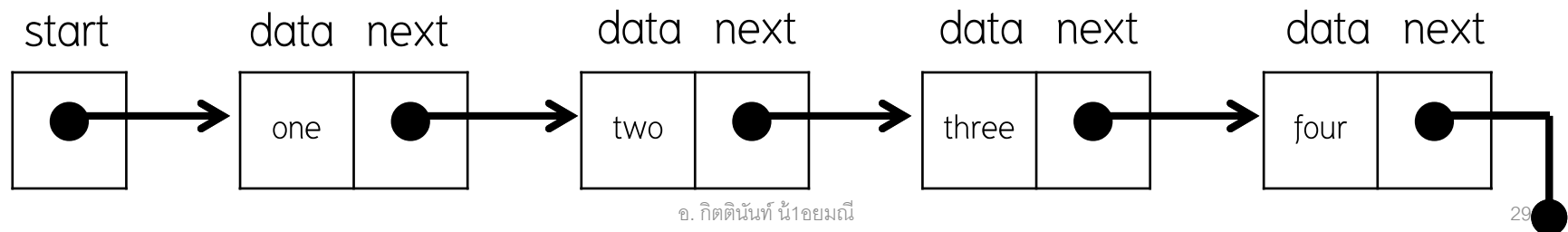
Simple Linked List

```
15 int main(){
16     struct studentNode *start, *now1, **now2;
17     start = new struct studentNode;
18     SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```



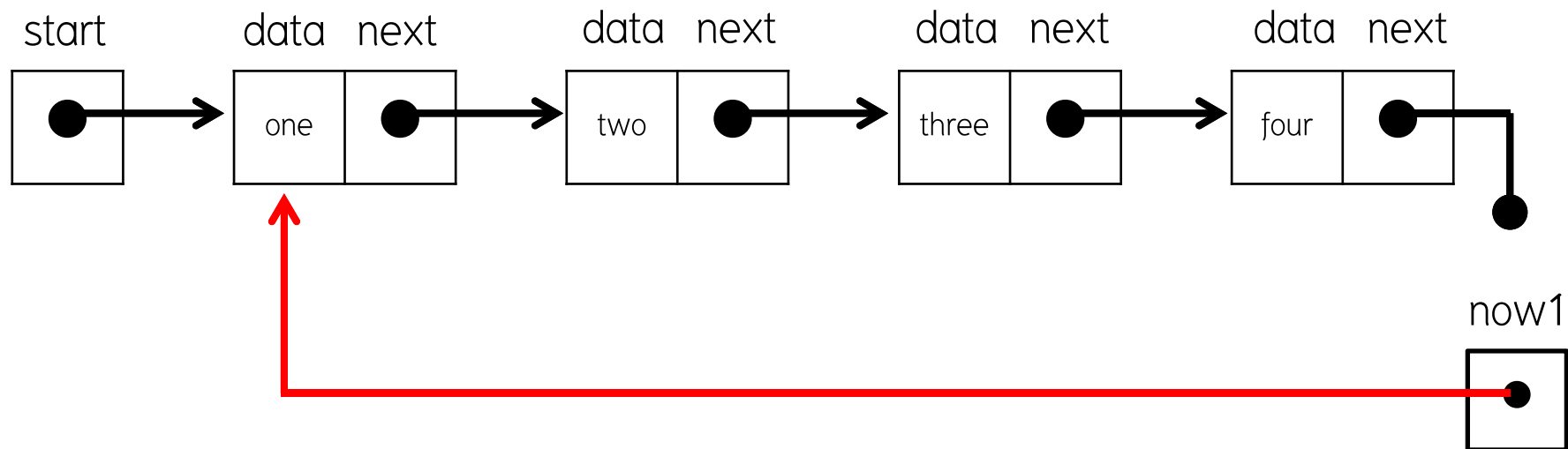
Simple Linked List

```
15 int main(){
16     struct studentNode *start, *now1, **now2;
17     start = new struct studentNode;
18     SaveNode(start, "one", 6, 'M', 3.11);
19     start->next = new struct studentNode;
20     SaveNode(start->next, "two", 8, 'F', 3.22);
21     start->next->next = new struct studentNode;
22     SaveNode(start->next->next, "three", 10, 'M', 3.33);
23     start->next->next->next = new struct studentNode;
24     SaveNode(start->next->next->next, "four", 12, 'F', 3.44);
25     now1 = start;
26     now2 = &start;
27     printf("%s\n", now1->name);
28     return 0;
29 }
```



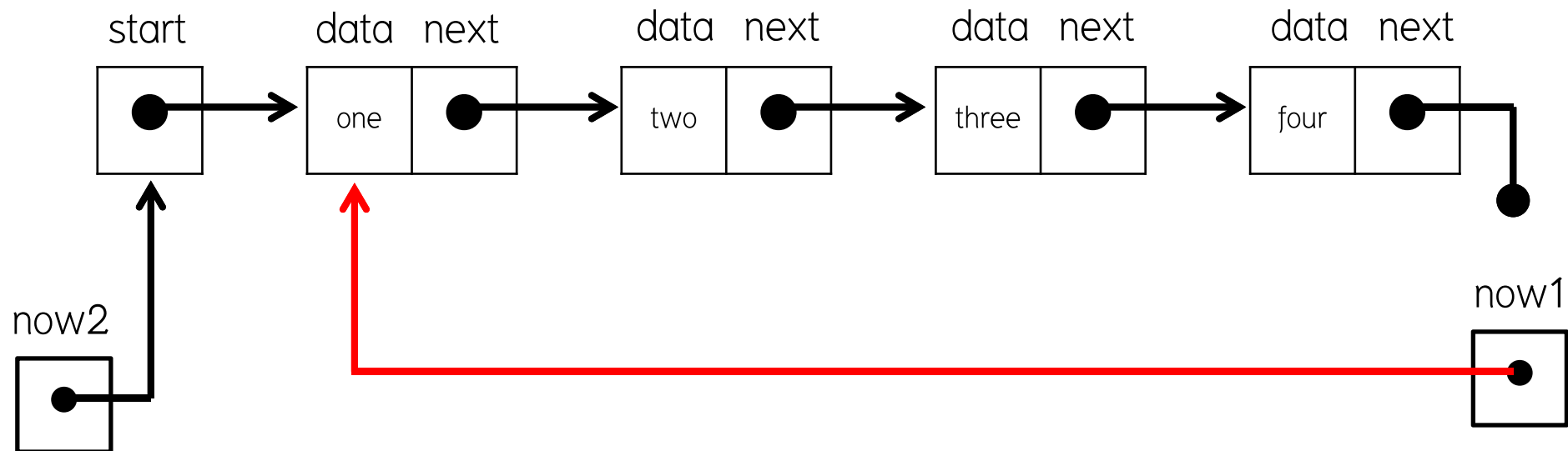
Simple Linked List

```
23 | start->next->next->next = new struct studentNode;  
24 | SaveNode(start->next->next->next, "four", 12, 'F', 3.44);  
25 | → now1 = start;  
26 | now2 = &start;  
27 | printf("%s\n", now1->name);  
28 | return 0;  
29 | }
```




Simple Linked List

```
23 start->next->next->next = new struct studentNode;  
24 SaveNode(start->next->next->next, "four", 12, 'F', 3.44);  
25 now1 = start;  
26 → now2 = &start;  
27 printf("%s\n", now1->name);  
28 return 0;  
29 }
```



Simple Linked List

```
28 |  
29 | printf("%s\n", now1->name);  
30 | printf("%s\n", (*now2)->name);  
31 | return 0;  
   | }
```



one
one

Simple Linked List

- นั่นคือ ถ้า Parameter เป็น Pointer ระดับเดียวกัน Pointer ภายนอก Function จะไม่สามารถเปลี่ยนที่ชี้ของภายนอกได้ ดังนั้น Parameter จะต้องเป็น Pointer ที่สูงกว่า 1 ระดับ
- ควรระวังเรื่องลำดับความสำคัญของเครื่องหมาย โดยใช้ วงเล็บ “(” และ “)” ครอบเอาไว้ให้ถูกต้อง